

EXHIBIT F

View Menu

```
Private Sub Form_Initialize()  
    Dim ItemIndex As Integer  
    Dim i As Integer  
    Dim Item As ListImage  
    Dim v As Variant  
    Dim c As Collection  
    Dim strTemp As String  
    Dim LoadParentEntity As Boolean  
    Dim LoadAdHocEntity As Boolean  
    Dim LoadReportEntity As Boolean  
    Dim LoadReferenceTable As Boolean  
  
    On Error GoTo ErrForm_Initialize  
  
    LoadParentEntity = (ParentCount > 0)  
    LoadReportEntity = (ReportCount > 0)  
    LoadReferenceTable = (ReferenceTableCount > 0)  
    LoadAdHocEntity = (ReportEntityCount > 0)  
  
    If LoadParentEntity Then  
        Set c = GetEntityIcons()  
        If Not c Is Nothing Then  
            i = 1  
            For Each v In c  
                strTemp = "KEY_" & v.EntityID  
                Set Item = imgLargeIcons.ListImages.Add(i, strTemp, LoadPicture(App.Path & "\" & gData.IconPath & "\" &  
v.IconFileName))  
                Item.Tag = v.EntityCaption  
                i = i + 1  
            Next v  
            Set c = Nothing  
        End If  
    End If  
    Me.Caption = gData.Caption  
    Me.Icon = LoadPicture(gData.ApplicationIcon)  
    Call SetAppIcon(Me)  
    Set Frame.ImageListControl = imgLargeIcons  
    ItemIndex = 0  
    m_FormLoaded = False  
    For i = 1 To imgLargeIcons.ListImages.Count  
        Set Item = imgLargeIcons.ListImages(i)  
        If (Item.Key = "REFERENCE_TABLE") Then  
            If LoadReferenceTable Then  
                Frame.AddItem Item.Key, Item.Tag, Item.Index, "pnlMaintenance"  
                If ItemIndex > 0 Then  
                    Load mnuViewItems(ItemIndex)  
                    mnuViewItems(ItemIndex).Caption = "-"  
                    ItemIndex = ItemIndex + 1  
                    Load mnuViewItems(ItemIndex)  
                End If  
                mnuViewItems(ItemIndex).Caption = "&" & Item.Tag  
                mnuViewItems(ItemIndex).Tag = Item.Key  
                ItemIndex = ItemIndex + 1  
            End If  
        ElseIf (Item.Key = "QUERY") Then  
            If LoadAdHocEntity Then  
                Frame.AddItem Item.Key, Item.Tag, Item.Index, "pnlEntityReports"  
                If ItemIndex > 0 Then  
                    Load mnuViewItems(ItemIndex)  
                    mnuViewItems(ItemIndex).Caption = "-"  
                    ItemIndex = ItemIndex + 1  
                    Load mnuViewItems(ItemIndex)  
                End If  
                mnuViewItems(ItemIndex).Caption = "&" & Item.Tag  
  
                mnuViewItems(ItemIndex).Tag = Item.Key  
                ItemIndex = ItemIndex + 1  
            End If  
        ElseIf (Item.Key = "REPORT") Then  
            If LoadReportEntity Then  
                Frame.AddItem Item.Key, Item.Tag, Item.Index, "pnlReportTool"  
                If ItemIndex > 0 Then  
                    Load mnuViewItems(ItemIndex)  
                    mnuViewItems(ItemIndex).Caption = "-"  
                    ItemIndex = ItemIndex + 1  
                    Load mnuViewItems(ItemIndex)  
                End If  
            End If  
        End If  
    Next i  
End Sub
```

EXHIBIT F

```
mnuViewItems(ItemIndex).Caption = "&" & Item.Tag
mnuViewItems(ItemIndex).Tag = Item.Key
ItemIndex = ItemIndex + 1
End If
Else
If LoadParentEntity Then
Frame.AddItem Right$(Item.Key, Len(Item.Key) - 4), Item.Tag, Item.Index, "pnlEntityView"
If ItemIndex > 0 Then
Load mnuViewItems(ItemIndex)
End If
mnuViewItems(ItemIndex).Caption = "&" & Item.Index & "." & Item.Tag
mnuViewItems(ItemIndex).Tag = Right(Item.Key, Len(Item.Key) - 4)
ItemIndex = ItemIndex + 1
End If
End If
Next i
If LoadAdHocEntity Then
pnlEntityReports.SetAppInformationObject gData.ApplicationInfo
pnlEntityReports.InitObject
End If
If LoadReportEntity Then
pnlReportTool.SetAppInformationObject gData.ApplicationInfo
pnlReportTool.InitObject
End If
Set Operation = New clsOperation
LoadTools
Exit Sub
ErrForm_Initialize:
HandleError Err

End Sub
```

EXHIBIT F

Parent Entities

```
Public Sub Refresh()  
    If m_CurrentEntityID = m_EntityID Then  
        grdResults.Refresh  
    ElseIf Filter = "" Then  
        m_CurrentEntityID = m_EntityID  
        LoadEntityParameters  
        txtCriteria.Text = ""  
        grdResults.EntityID = m_EntityID  
        grdResults.LoadStructureOnly = True  
        grdResults.CustomSQL = False  
        grdResults.Filter = ""  
        grdResults.Refresh  
    Else  
        m_CurrentEntityID = m_EntityID  
        LoadEntityParameters  
        txtCriteria.Text = ""  
        grdResults.EntityID = m_EntityID  
        grdResults.LoadStructureOnly = False  
        grdResults.CustomSQL = False  
        grdResults.Filter = Filter  
        grdResults.Refresh  
    End If  
    SetCurrentID  
  
    On Error Resume Next  
    txtCriteria.SetFocus  
    Err.Clear  
End Sub
```

```
Private Sub SetCurrentID()  
    Static v As Variant  
  
    LockWindow UserControl.Parent.hwnd, True  
    If grdResults.RecordCount = 0 Then  
        v = -1  
    Else  
        v = grdResults.RowID  
    End If  
    EntityRelations.ShowAllChildren = True  
    EntityRelations.ParentRowID = v  
    EntityRelations.ParentEntityID = m_EntityID  
    v = EntityRelations.ChildRowID  
    EntityRelations.Refresh  
    EntityRelations.ChildRowID = v  
    On Error Resume Next  
    grdResults.SetFocus  
    Err.Clear  
    LockWindow UserControl.Parent.hwnd, False  
End Sub
```

(From UserInterface\Interface\Source\GenericControls\ListBoxControl.ctl)

```
Public Sub Refresh()  
    Dim v As Variant  
    Dim CR As COMMAND_T  
    Dim i As Long  
  
    m>LoadingControl = True  
    AllowDataEntry = False  
    LockWindow CurrentGrid().hwnd, True  
  
    If m_ParentEntityID = -1 Then  
        m_RelationID = -1  
        If GetAppInfo(m_EntityID, "ENTITY", "REFERENCE") Then  
            m_Context = "REFERENCE"  
        Else  
            m_Context = "ENTITY"  
        End If  
    Else  
        m_RelationID = gData.ApplicationInfo.GetRelationshipInfo(m_ParentEntityID, m_EntityID, "RELATION_ID")  
        m_Context = gData.ApplicationInfo.GetRelationshipInfo(m_ParentEntityID, m_EntityID, "RELATION_TYPE")  
    End If  
  
    If m_ObjectID = -1 Then m_ObjectID = MessageHandler.GetNextObjectID()  
    If m_RelationID = -1 Then  
        v = GetAppInfo(m_EntityID, "ENTITY_ATTRIBUTE", "ALLOW_GROUP_BY")  
    Else  
        v = GetAppInfo(m_RelationID, "RELATIONSHIP_ATTRIBUTE", "ALLOW_GROUP_BY")  
    End If
```

EXHIBIT F

```
End If
If IsNull(v) Then v = 0
CurrentGrid().GroupByBoxVisible = CBool(v)

Set m_SearchColumns = Nothing
Select Case m_Context
    Case "ENTITY"
        Set m_SearchColumns = GetSearchColumns(m_EntityID, False)
    Case "REFERENCE"
    Case Else
        Set m_SearchColumns = GetSearchColumns(m_RelationID, True)
End Select

m_KeyField = GetAppInfo(m_EntityID, "ENTITY", "UNIQUE_FIELD_NAME")
m_ShowKeyField = GetAppInfo(m_EntityID, "ENTITY", "SHOW_UNIQUE_FIELD")
Select Case m_Context
    Case "MANY TO MANY"
        m_MapsToKeyField = GetAppInfo(gData.ApplicationInfo.GetMapsToEntityID(m_ParentEntityID, m_EntityID),
"ENTITY", "UNIQUE_FIELD_NAME")
    Case Else
        m_MapsToKeyField = ""
End Select
Select Case m_Context
    Case "ENTITY", "REFERENCE"
        m_FilterField = m_KeyField
        m_ShowFilterField = m_ShowKeyField
    Case Else
        m_FilterField = GetAppInfo(m_ParentEntityID, "ENTITY", "UNIQUE_FIELD_NAME") & ""
        m_ShowFilterField = False
End Select

LockWindow CurrentGrid().hwnd, False
LoadGrid
LockWindow CurrentGrid().hwnd, True
If Not CurrentGrid().Visible Then CurrentGrid().Visible = True
m_RowID = -1
SetCurrentRow
EnableMenus
m_SortColumn = ""
LockWindow CurrentGrid().hwnd, False
m>LoadingControl = False
End Sub
```

EXHIBIT F

Quick-Text Search

```
Private Sub RunQuery()  
    Dim SelectClause As String  
    Dim OrderByClause As String  
    Dim RecordSource As String  
  
    LockWindow UserControl.Parent.hwnd, True  
    RecordSource = GetAppInfo(m_EntityID, "ENTITY", "RECORD_SOURCE")  
    OrderByClause = gData.ApplicationInfo.GetEntityOrderBy(m_EntityID)  
    If OrderByClause <> "" Then  
        OrderByClause = "Order By " & OrderByClause  
    End If  
    grdResults.Filter = ""  
    grdResults.LoadStructureOnly = False  
    grdResults.CustomSQL = True  
    m_SelectAllSymbol = GetAppInfo("SELECT_ALL_SYMBOL", "SYSTEM_INFO", "ATTRIBUTE_VALUE") & ""  
    SelectClause = "Select " & RecordSource & ". * From " & RecordSource & " "  
    If Trim$(txtCriteria.Text) = m_SelectAllSymbol Then  
        grdResults.SQL = SelectClause & OrderByClause  
    Else  
        grdResults.SQL = SearchEntity(m_EntityID, txtCriteria.Text)  
    End If  
  
    LockWindow UserControl.Parent.hwnd, False  
    grdResults.Refresh  
    RecordCountMessage  
    On Error Resume Next  
    txtCriteria.SetFocus  
    Err.Clear  
End Sub  
  
Public Function SearchEntity(EntityID As Long, SearchValue As String) As String  
    Dim EntityChildren As Collection  
    Dim SearchColumns As Collection  
    Dim SelectClause As String  
    Dim WhereClause As String  
    Dim PrimarySelectClause As String  
    Dim PrimaryWhereClause As String  
    Dim PrimaryOrderByClause As String  
    Dim PrimaryKeyField As String  
    Dim PrimaryTable As String  
    Dim KeyField As String  
    Dim TempWhereClause As String  
    Dim ERD() As ENTITY_RELATION_T  
    Dim ER As ENTITY_RELATION_T  
    Dim EC As ENTITY_COLUMN_T  
    Dim rs As ADODB.Recordset  
    Dim fDef As ADODB.Field  
    Dim i As Long  
    Dim j As Long  
    Dim IgnoreColumn As Boolean  
    Dim AllowSearch As Boolean  
    Dim SearchType As String  
    Dim v As Variant  
    Dim Names() As String  
    Dim s As String  
    Dim SQL As String  
    Dim DatasourceID As Long  
    Dim PlatformID As Long  
    Dim RightFieldDelimiter As String  
    Dim LeftFieldDelimiter As String  
  
    DatasourceID = GetAppInfo(EntityID, "ENTITY", "DATASOURCE_ID")  
    PlatformID = GetAppInfo(DatasourceID, "DATASOURCE", "PLATFORM_ID")  
  
    Select Case GetAppInfo(PlatformID, "PLATFORM_ATTRIBUTE", "FIELD_DELIMITER")  
        Case "BRACKET"  
            LeftFieldDelimiter = "["  
            RightFieldDelimiter = "]"  
        Case "DOUBLE_QUOTE"  
            LeftFieldDelimiter = Chr(34)  
            RightFieldDelimiter = Chr(34)  
    End Select  
  
    Set EntityChildren = gData.ApplicationInfo.GetEntityChildren(EntityID)  
  
    ERD = gData.ApplicationInfo.GetERD()  
    For i = 1 To UBound(ERD)
```

EXHIBIT F

```

ER = ERD(i)
WhereClause = ""
KeyField = GetAppInfo(ER.EntityID, "ENTITY", "UNIQUE_FIELD_NAME") & ""
If ER.EntityID = EntityID Then
    PrimaryKeyField = KeyField
    PrimaryTable = ER.TableName
    s = GetAppInfo(ER.EntityID, "ENTITY", "RECORD_SOURCE")
    PrimarySelectClause = "SELECT " & s & ". * FROM " & GetTablePrefix(DatasourceID) & s & " "
    PrimaryOrderByClause = gData.ApplicationInfo.GetEntityOrderBy(ER.EntityID)
    If PrimaryOrderByClause <> "" Then
        PrimaryOrderByClause = "ORDER BY " & PrimaryOrderByClause
    End If
    SelectClause = PrimarySelectClause
Else
    SelectClause = "SELECT " & KeyField & " FROM " & GetTablePrefix(DatasourceID) & GetAppInfo(ER.EntityID,
"ENTITY", "RECORD_SOURCE") & " "
End If

If EntityID = ER.EntityID Then
    AllowSearch = True
Else
    On Error Resume Next
    v = EntityChildren("ENTITY_" & ER.EntityID)
    If Err.Number = 0 Then
        AllowSearch = True
    Else
        AllowSearch = False
        Err.Clear
    End If
End If
If Not AllowSearch Then
    ER.Tag = ""
Else
    Set rs = GetEmptyRS(ER.EntityID, False)
    If rs Is Nothing Then
        WhereClause = "WHERE 1=2 "
        SearchEntity = PrimarySelectClause & WhereClause
        Exit Function
    End If
    If (EntityID = ER.EntityID) Then
        Set SearchColumns = gData.ApplicationInfo.GetSearchColumns(ER.EntityID, False)
    Else
        Set SearchColumns = gData.ApplicationInfo.GetSearchColumns(v.RelationID, True)
    End If

    For Each fDef In rs.Fields
        IgnoreColumn = False
        If SearchColumns Is Nothing And (EntityID = ER.EntityID) Then
            SearchType = "BEGINS"
        Else
            On Error Resume Next
            EC = SearchColumns(fDef.Name)
            If Err.Number = 0 Then
                SearchType = EC.SearchType
            Else
                IgnoreColumn = True
                Err.Clear
            End If
        End If
        If Not IgnoreColumn Then
            Select Case GetGenericDatatype(fDef.Type)
                Case GenericText
                    If WhereClause <> "" Then WhereClause = WhereClause & " OR "
                    If SearchType = "BEGINS" Then
                        WhereClause = WhereClause & LeftFieldDelimiter & fDef.Name & RightFieldDelimiter & " Like " &
FixSQLValue(ER.EntityID, SearchValue, Null, GenericText, oeBEGINS)
                    ElseIf SearchType = "INCLUDES" Then
                        WhereClause = WhereClause & LeftFieldDelimiter & fDef.Name & RightFieldDelimiter & " Like " &
FixSQLValue(ER.EntityID, SearchValue, Null, GenericText, oeINCLUDES)
                    ElseIf SearchType = "NAME_SEARCH" Then
                        ParseName SearchValue, Names()
                        s = ""
                        For j = 0 To UBound(Names)
                            If s <> "" Then s = s & " And "
                            s = s & LeftFieldDelimiter & fDef.Name & RightFieldDelimiter & " Like " &
FixSQLValue(ER.EntityID, Names(j), Null, GenericText, oeINCLUDES)
                        Next j
                        If s <> "" Then
                            s = "(" & s & ")"
                        End If
                    End Case
                Case GenericNumber
                    If SearchType = "BEGINS" Then
                        WhereClause = WhereClause & LeftFieldDelimiter & fDef.Name & RightFieldDelimiter & " Like " &
FixSQLValue(ER.EntityID, SearchValue, Null, GenericText, oeBEGINS)
                    ElseIf SearchType = "INCLUDES" Then
                        WhereClause = WhereClause & LeftFieldDelimiter & fDef.Name & RightFieldDelimiter & " Like " &
FixSQLValue(ER.EntityID, SearchValue, Null, GenericText, oeINCLUDES)
                    End Case
            End Select
        End If
    Next fDef
End For

```

EXHIBIT F

```
        WhereClause = WhereClause & s
    End If
Else
    WhereClause = WhereClause & LeftFieldDelimiter & fDef.Name & RightFieldDelimiter & " = " &
FixSQLValue(ER.EntityID, SearchValue, Null, GenericText, oeEQ)
End If
Case GenericDate
    If IsDate(SearchValue) Then
        If WhereClause <> "" Then WhereClause = WhereClause & " OR "
        WhereClause = WhereClause & LeftFieldDelimiter & fDef.Name & RightFieldDelimiter & " = " &
FixSQLValue(ER.EntityID, SearchValue, Null, GenericDate, oeEQ)
    End If
Case GenericNumber
    If IsNumeric(SearchValue) Then
        If WhereClause <> "" Then WhereClause = WhereClause & " OR "
        WhereClause = WhereClause & LeftFieldDelimiter & fDef.Name & RightFieldDelimiter & " = " &
FixSQLValue(ER.EntityID, SearchValue, Null, GenericNumber, oeEQ)
    End If
End Select
End If
Next fDef
End If
If WhereClause = "" Then
    ER.Tag = ""
    SQL = ""
Else
    WhereClause = "WHERE " & WhereClause & " "
    SQL = SelectClause & WhereClause
End If
If ER.EntityID = EntityID Then
    PrimaryWhereClause = WhereClause
End If
ER.Tag = SQL
ERD(i) = ER
Next i

SQL = ""
WhereClause = ""
For i = 1 To UBound(ERD)
    ER = ERD(i)
    If Not GetAppInfo(ER.EntityID, "ENTITY", "REFERENCE") Then
        If (PrimaryTable <> ER.TableName) And (ER.Tag <> "") Then
            If BuildSQL(EntityID, "", PrimaryTable, ER.TableName, ER.Tag) Then
                If WhereClause <> "" Then WhereClause = WhereClause & " AND "
                WhereClause = WhereClause & PrimaryKeyField & " IN (" & ER.Tag & ")"
            End If
        End If
    End If
End If
Next i
If WhereClause = "" Then
    WhereClause = PrimaryWhereClause
Else
    If PrimaryWhereClause = "" Then
        WhereClause = " WHERE " & WhereClause & " "
    Else
        WhereClause = PrimaryWhereClause & " OR " & WhereClause & " "
    End If
End If
SQL = PrimarySelectClause & WhereClause & PrimaryOrderByClause
SearchEntity = SQL
End Function
```

EXHIBIT F

Advanced Searching

```
Private Function SQL() As String
    Dim RUtils As clsReportUtils
    Dim SelectClause As String
    Dim WhereClause As String
    Dim OrderByClause As String
    Dim FromClause As String
    Dim PrimarySelectClause As String
    Dim PrimaryWhereClause As String
    Dim PrimaryKeyField As String
    Dim PrimaryTable As String
    Dim RecordSource As String
    Dim PrimaryRecordSource As String
    Dim KeyField As String
    Dim TempWhereClause As String
    Dim ER As ENTITY_RELATION_T
    Dim i As Long
    Dim DatasourceID As Long

    WhereClause = ""
    FromClause = ""
    DatasourceID = GetAppInfo(m_EntityID, "ENTITY", "DATASOURCE_ID")
    For i = 1 To UBound(m_ERD)
        ER = m_ERD(i)
        If Not GetAppInfo(ER.EntityID, "ENTITY", "REFERENCE") Then
            RecordSource = GetAppInfo(ER.EntityID, "ENTITY", "RECORD_SOURCE")
            If ER.EntityID = m_EntityID Then

                PrimaryRecordSource = RecordSource
                KeyField = GetAppInfo(ER.EntityID, "ENTITY", "UNIQUE_FIELD_NAME") & ""
                PrimaryTable = ER.TableName
                PrimarySelectClause = "SELECT " & RecordSource & "." & KeyField & " FROM "
                If WhereClause <> "" Then WhereClause = " AND " & WhereClause
                WhereClause = RecordSource & "." & KeyField & "=" & PrimaryTable & "." & KeyField & WhereClause
                OrderByClause = gData.ApplicationInfo.GetEntityOrderBy(ER.EntityID)
                If OrderByClause <> "" Then
                    OrderByClause = " ORDER BY " & OrderByClause
                End If
                SelectClause = PrimarySelectClause
            Else
                Set RUtils = m_SearchTree("ENTITY_" & ER.EntityID)
                If Not (RUtils.FilterCriteriaRS.BOF And RUtils.FilterCriteriaRS.EOF) Then
                    ER.Tag = "CRITERIA DEFINED"
                    If FromClause <> "" Then FromClause = FromClause & ","
                    FromClause = FromClause & ER.TableName & "," & RecordSource
                End If
            End If
            m_ERD(i) = ER
        End If
    Next i
    For i = 1 To UBound(m_ERD)
        ER = m_ERD(i)
        If Not GetAppInfo(ER.EntityID, "ENTITY", "REFERENCE") Then
            If (PrimaryTable <> ER.TableName) And (ER.Tag <> "") Then
                If BuildSQL(m_EntityID, "", PrimaryTable, ER.TableName, WhereClause) Then

                    End If
                End If
            End If
        End If
    Next i

    PrimarySelectClause = PrimarySelectClause & BuildFromClause(WhereClause, GetTablePrefix(GetAppInfo(m_EntityID,
"ENTITY", "DATASOURCE_ID")))
    If m_Filter <> "" Then
        WhereClause = " WHERE " & WhereClause & " AND " & m_Filter & " AND " & TraverseTree(lstFilters,
lstFilters.Nodes(1).Child)
    Else
        WhereClause = " WHERE " & WhereClause & " AND " & TraverseTree(lstFilters, lstFilters.Nodes(1).Child)
    End If
    SQL = "SELECT " & PrimaryRecordSource & "." & KeyField & " FROM " & GetTablePrefix(DatasourceID) & PrimaryRecordSource & "
WHERE " & PrimaryRecordSource & "." & KeyField &
    & " IN (" & PrimarySelectClause & WhereClause & ")" & OrderByClause

End Function
```


EXHIBIT F

Child Entities

```
Public Sub Refresh()  
    Static CurrentEntityID As Long  
    Dim CurrentItem As Long  
  
    grdData.LoadStructureOnly = False  
    grdData.CustomSQL = False  
    If CurrentEntityID = m_ParentEntityID Then  
        If lstChildren.ListItems.Count = 0 Then  
            CurrentItem = -1  
        Else  
            CurrentItem = lstChildren.SelectedItem.Tag  
        End If  
    Else  
        CurrentEntityID = m_ParentEntityID  
        LoadSourceList m_ParentEntityID  
        CurrentItem = -1  
    End If  
    If lstChildren.ListItems.Count = 0 Then  
        grdData.ClearList  
        CurrentItem = -1  
        grdData.Enabled = False  
        lstChildren.BackColor = UserControl.BackColor  
  
    ElseIf m_ParentEntityID = -1 Then  
        Set lstChildren.SelectedItem = lstChildren.FindItem(CurrentItem, lvwTag)  
        grdData.Enabled = False  
    Else  
        If CurrentItem = -1 Then  
            Set lstChildren.SelectedItem = lstChildren.ListItems(1)  
        Else  
            Set lstChildren.SelectedItem = lstChildren.FindItem(CurrentItem, lvwTag)  
        End If  
  
        lstChildren_ItemClick lstChildren.SelectedItem  
        lstChildren.ForeColor = vbBlue  
        lstChildren.BackColor = vbWhite  
        grdData.Enabled = True  
    End If  
End Sub  
  
Private Sub LoadSourceList(EntityID As Long)  
    LoadEntityChildren EntityID, lstChildren, m_ShowAllChildren  
    ResizeListBox lstChildren  
    UserControl_Resize  
End Sub  
  
Public Sub LoadEntityChildren(ParentEntityID As Long, lstChildren As Object, ShowAllChildren As Boolean)  
    Dim c As Collection  
    Dim v As Variant  
    Dim Item As Object  
    Dim i As Long  
  
    If TypeName(lstChildren) = "ListView" Then  
        lstChildren.ListItems.Clear  
        Set c = AppInformation.GetEntityChildren(ParentEntityID)  
        For Each v In c  
            If (Not ShowAllChildren And Not v.DisplayOnTab) Or (ShowAllChildren) Then  
                If v.EntityType = "MANY_TO_MANY" Then  
                    i = AppInformation.GetMapsToEntityID(ParentEntityID, v.EntityID)  
                    Set Item = lstChildren.ListItems.Add(, , AppInformation.GetAppInfoValue(i, "ENTITY",  
"ENTITY_CAPTION"))  
                Else  
                    Set Item = lstChildren.ListItems.Add(, , v.EntityCaption)  
                End If  
                Item.Tag = v.EntityID  
                If v.Required Then  
                    Item.ForeColor = vbRed  
                End If  
            End If  
        Next v  
        Set c = Nothing  
    ElseIf TypeName(lstChildren) = "ListBox" Or TypeName(lstChildren) = "ComboBox" Then  
        lstChildren.Clear  
        Set c = AppInformation.GetEntityChildren(ParentEntityID)  
        For Each v In c  
            If (Not ShowAllChildren And Not v.DisplayOnTab) Or (ShowAllChildren) Then  
                lstChildren.AddItem v.EntityCaption  
            End If  
        Next v  
    End If  
End Sub
```

EXHIBIT F

```
    lstChildren.ItemData(lstChildren.NewIndex) = v.EntityID
End If
Next v
Set c = Nothing
End If
End Sub

Private Sub lstChildren_ItemClick(ByVal Item As MSComctlLib.ListItem)
    If (lstChildren.ListItems.Count = 0) Or (lstChildren.SelectedItem Is Nothing) Then
        Exit Sub
    End If
    UserControl.Enabled = False
    m_ChildEntityID = Item.Tag
    LoadRelatedEntity m_ChildEntityID
    grdData.RowID = -1
    grdData.Refresh
    UserControl.Enabled = True
    On Error Resume Next
    lstChildren.SetFocus
    Err.Clear
End Sub

Private Sub LoadRelatedEntity(ChildEntityID As Long)
    m_Context = gData.ApplicationInfo.GetRelationshipInfo(m_ParentEntityID, m_ChildEntityID, "RELATION_TYPE")
    grdData.Filtered = False
    grdData.ParentRowID = m_ParentRowID
    grdData.ParentEntityID = m_ParentEntityID
    grdData.EntityID = m_ChildEntityID
    grdData.Context = m_Context
    grdData.CustomSQL = False
    Select Case m_Context
        Case "ONE_TO_MANY", "MANY_TO_MANY"
            grdData.Filter = GetAppInfo(m_ChildEntityID, "ENTITY", "TABLE_NAME") & "." & _
                gData.ApplicationInfo.GetRelationshipInfo(m_ParentEntityID, m_ChildEntityID, "RELATED_FIELD_NAME") & " = " & m_ParentRowID
        Case "MANY_TO_ONE"
            grdData.Filter = GetAppInfo(m_ParentEntityID, "ENTITY", "TABLE_NAME") & "." & _
                GetAppInfo(m_ParentEntityID, "ENTITY", "UNIQUE_FIELD_NAME") & " = " & m_ParentRowID
    End Select
End Sub

(From D:\UserInterface\AppInfo\clsAppInfo.cls)
Public Function GetEntityChildren(ParentEntityID As Long) As Collection
    Dim c As Collection
    Dim ED As ENTITY_DATA_T
    Dim i As Long

    Set c = New Collection
    With m_rsEntityRelationship
        .Filter = "VISIBLE=TRUE AND REFERENCE=FALSE AND ENTITY_ID=" & ParentEntityID
        On Error Resume Next
        .Sort = "RELATIVE_ORDER ASC"
        Err.Clear
        Do While Not .EOF
            ED.EntityID = !RELATED_ENTITY_ID
            ED.RelationID = !RELATION_ID
            ED.EntityCaption = Trim$(!ENTITY_CAPTION)
            ED.EntityType = !RELATION_TYPE
            ED.Required = !Required
            ED.DisplayOnTab = !DISPLAY_ON_TAB
            c.Add ED, "ENTITY_" & ED.EntityID
            .MoveNext
        Loop
        .Filter = adFilterNone
        .MoveFirst
    End With
    Set GetEntityChildren = c
    Set c = Nothing
End Function
```

EXHIBIT F

Menus and Menu Contexts

```
Public Sub Refresh()
    Dim v As Variant
    Dim CR As COMMAND_T
    Dim i As Long

    m>LoadingControl = True
    AllowDataEntry = False
    LockWindow CurrentGrid().hwnd, True

    If m_ParentEntityID = -1 Then
        m_RelationID = -1
        If GetAppInfo(m_EntityID, "ENTITY", "REFERENCE") Then
            m_Context = "REFERENCE"
        Else
            m_Context = "ENTITY"
        End If
    Else
        m_RelationID = gData.ApplicationInfo.GetRelationshipInfo(m_ParentEntityID, m_EntityID, "RELATION_ID")
        m_Context = gData.ApplicationInfo.GetRelationshipInfo(m_ParentEntityID, m_EntityID, "RELATION_TYPE")
    End If

    If m_ObjectID = -1 Then m_ObjectID = MessageHandler.GetNextObjectID()
    If m_RelationID = -1 Then
        v = GetAppInfo(m_EntityID, "ENTITY_ATTRIBUTE", "ALLOW_GROUP_BY")
    Else
        v = GetAppInfo(m_RelationID, "RELATIONSHIP_ATTRIBUTE", "ALLOW_GROUP_BY")
    End If
    If IsNull(v) Then v = 0
    CurrentGrid().GroupByBoxVisible = CBool(v)

    Set m_SearchColumns = Nothing
    Select Case m_Context
        Case "ENTITY"
            Set m_SearchColumns = GetSearchColumns(m_EntityID, False)
        Case "REFERENCE"
        Case Else
            Set m_SearchColumns = GetSearchColumns(m_RelationID, True)
    End Select

    m_KeyField = GetAppInfo(m_EntityID, "ENTITY", "UNIQUE_FIELD_NAME")
    m_ShowKeyField = GetAppInfo(m_EntityID, "ENTITY", "SHOW_UNIQUE_FIELD")
    Select Case m_Context
        Case "MANY_TO_MANY"
            m_MapstoKeyField = GetAppInfo(gData.ApplicationInfo.GetMapsToEntityID(m_ParentEntityID, m_EntityID),
"ENTITY", "UNIQUE_FIELD_NAME")
        Case Else
            m_MapstoKeyField = ""
    End Select
    Select Case m_Context
        Case "ENTITY", "REFERENCE"
            m_FilterField = m_KeyField
            m_ShowFilterField = m_ShowKeyField
        Case Else
            m_FilterField = GetAppInfo(m_ParentEntityID, "ENTITY", "UNIQUE_FIELD_NAME") & ""
            m_ShowFilterField = False
    End Select

    LockWindow CurrentGrid().hwnd, False
    LoadGrid
    LockWindow CurrentGrid().hwnd, True
    If Not CurrentGrid().Visible Then CurrentGrid().Visible = True
    m_RowID = -1
    SetCurrentRow
    EnableMenus
    m_SortColumn = ""
    LockWindow CurrentGrid().hwnd, False
    m>LoadingControl = False
End Sub

Private Sub EnableMenus()
    On Error Resume Next
    Select Case CurrentGrid().Name
        Case "lstData" 'List View
            m_DefaultMenuItem = LoadMenuItems(m_Context, "LIST", mnuOptionsDetail, VitemArray)
        Case "grdData" 'Grid View
            m_DefaultMenuItem = LoadMenuItems(m_Context, "GRID", mnuOptionsDetail, VitemArray)
```

EXHIBIT F

```
End Select

Select Case m_Context
    Case "ENTITY", "REFERENCE"
        DisableMenuOptions m_EntityID, gData.GroupID
    Case Else
        DisableMenuOptions m_RelationID, gData.GroupID
End Select
Err.Clear
End Sub
```

EXHIBIT F

Data Entry Forms

```
Option Explicit
Private m_Cancel As Boolean
Private m_DataEntryControl As Object
Private m_FormLoaded As Boolean
Public Message As MessageClass
Private WithEvents Operation As clsOperation
Private m_DefaultMenuItem As Long
Private VItemArray As Variant
Private m_Operations As Collection
Public CallUpdateRecordOperation As Boolean
Private m_Script As ScriptControl
Private m_TableName As String
Private m_EntityID As Long
Private m_AppEnvironment As CAppEnvironment
Private m_CancelLoad As Boolean
Private Sub cmdCancel_Click()
    Message.MessageType = MTCancel
    m_Cancel = True
    DoUnload
End Sub

Private Sub cmdOK_Click()
    Dim ChildID As Long
    Dim i As Long
    Dim Msg As String
    Dim Cancel As Integer

    If Message.MessageType <> MTViewRecord Then
        RaiseEventEx "BeforeValidate", m_TableName, Cancel
        If Cancel Then Exit Sub
        If Not m_DataEntryControl.ValidData() Then Exit Sub
        RaiseEventEx "AfterValidate", m_TableName, Cancel
        If Cancel Then Exit Sub
        SaveEntity Cancel
        If Cancel Then Exit Sub
    End If
    If Message.Context = "ENTITY" Then
        If Message.MessageType <> MTViewRecord Then
            ChildID = RequiredChildren(Message.EntityID, CLng(Message.RowIDArray(0)))
            If ChildID <> -1 Then
                For i = 1 To tabView.Tabs.Count
                    Select Case tabView.Tabs(i).Key
                        Case "General", "RelatedInfo"
                        Case Else
                            If CLng(Right$(tabView.Tabs(i).Key, Len(tabView.Tabs(i).Key) - 1)) = ChildID Then
                                Msg = "Not all recommended data entry has been completed. Would you like to complete the recommended data entry now?"
                                If MsgBox(Msg, vbYesNo + vbQuestion, Me.Caption) = vbYes Then
                                    tabView.Tabs(i).Selected = True
                                    Exit Sub
                                End If
                            End If
                        End Select
                    End If
                Next i
            End If
        End If
        m_Cancel = False
        DoUnload
    End Sub

Private Sub Form_Activate()
    On Error Resume Next
    m_DataEntryControl.SetFocus
    Err.Clear
    If m_CancelLoad Then cmdCancel_Click
End Sub

Private Sub SetEntityObject()
    Dim LocalEntityID As Long
    Dim LocalRowID As Long
    Dim ParentKeyField As String
    Dim ControlName As String

    Select Case Message.Context
```

EXHIBIT F

```
Case "MANY_TO_MANY"
    LocalEntityID = gData.ApplicationInfo.GetMapsToEntityID(Message.ParentEntityID, Message.EntityID)
    LocalRowID = GetMapsToRowID(LocalEntityID, Message.EntityID, CLng(Message.RowIDArray(0)))
Case Else
    LocalEntityID = Message.EntityID
    LocalRowID = Message.RowIDArray(0)
End Select
m_EntityID = LocalEntityID
m_TableName = GetAppInfo(LocalEntityID, "ENTITY", "TABLE_NAME") & ""
ControlName = GetAppInfo(LocalEntityID, "ENTITY_ATTRIBUTE", "USERCONTROL") & ""
If ControlName = "" Then
    Set m_DataEntryControl = Controls.Add("AppSwift.DataEntryControl", "DEControl")
    m_DataEntryControl.Context = Message.Context
Else
    On Error Resume Next
    Set m_DataEntryControl = Controls.Add(gData.DataEntryControl & "." & ControlName, "DEControl")
    If Err.Number <> 0 Then
        Err.Clear
        Set m_DataEntryControl = Controls.Add("AppSwift.DataEntryControl", "DEControl")
        m_DataEntryControl.Context = Message.Context
    End If
End If

m_DataEntryControl.EntityID = LocalEntityID
m_DataEntryControl.SetAppInformationObject gData.ApplicationInfo

On Error Resume Next
Set m_DataEntryControl.MessageHandler = MessageHandler
Err.Clear

On Error GoTo ErrSetEntityObject

LoadEntityTabs LocalEntityID, tabView

If Message.MessageType = MTDuplicateRecord Then
    m_DataEntryControl.NewRecord = True
ElseIf Message.MessageType = MTNewRecord Then
    m_DataEntryControl.NewRecord = True
ElseIf Message.MessageType = MTViewRecord Then
    m_DataEntryControl.ReadOnly = True
    m_DataEntryControl.NewRecord = False
    cmdOK.Caption = "&Close"
    cmdCancel.Visible = False
    tbarOptions.Enabled = True
ElseIf Message.MessageType = MTEditRecord Then
    m_DataEntryControl.NewRecord = False
End If

Set m_DataEntryControl.rsData = Message.CurrentRS
If TypeName(m_DataEntryControl) = "DataEntryControl" And (Message.Context = "ONE_TO_MANY" Or Message.Context = "MANY_TO_MANY") Then
    On Error Resume Next
    ParentKeyField = GetAppInfo(Message.ParentEntityID, "ENTITY", "UNIQUE_FIELD_NAME")
    m_DataEntryControl.ParentEntityID = Message.ParentEntityID
    m_DataEntryControl.SetParentID ParentKeyField, Message.ParentRowID
    Err.Clear
End If
m_DataEntryControl.Refresh

Me.Caption = GetAppInfo(LocalEntityID, "ENTITY", "ENTITY_CAPTION") & " Input"
Select Case Message.MessageType
Case MTViewRecord
Case MTDuplicateRecord, MTNewRecord
    m_DataEntryControl.SetCurrentID "New Record"
    If (Message.Context = "ONE_TO_MANY" Or Message.Context = "MANY_TO_MANY") Then
        On Error Resume Next
        ParentKeyField = GetAppInfo(Message.ParentEntityID, "ENTITY", "UNIQUE_FIELD_NAME")
        m_DataEntryControl.SetParentID ParentKeyField, Message.ParentRowID
        Err.Clear
    End If
Case MTEditRecord
    If (Message.Context = "ONE_TO_MANY" Or Message.Context = "MANY_TO_MANY") Then
        On Error Resume Next
        ParentKeyField = GetAppInfo(Message.ParentEntityID, "ENTITY", "UNIQUE_FIELD_NAME")
        m_DataEntryControl.SetParentID ParentKeyField, Message.ParentRowID
        Err.Clear
    End If
End Select
ERelations.ShowAllChildren = False
```

EXHIBIT F

```
ERelations.ParentEntityID = LocalEntityID
ERelations.ParentRowID = LocalRowID
ERelations.Refresh
If ERelations.ListCount > 0 Then
    tabView.Tabs.Add , "RelatedInfo", "&Related Info"
End If
Exit Sub
ErrSetEntityObject:
    HandleError Err
    Err.Clear
End Sub

Private Sub SetFormSize()
    Dim BorderHeight As Single
    Dim BorderWidth As Single
    Dim MinimumHeight As Single
    Dim MinimumWidth As Single
    Dim DEControlHeight As Long
    Dim DEControlWidth As Long

    If Me.WindowState = vbMinimized Or Me.WindowState = vbMaximized Then Exit Sub

    On Error Resume Next

    BorderHeight = Me.Height - Me.ScaleHeight + 75
    BorderWidth = Me.Width - Me.ScaleWidth + 75
    If TypeName(m_DataEntryControl) = "DataEntryControl" Then
        m_DataEntryControl.GetCardSize DEControlHeight, DEControlWidth
        m_DataEntryControl.Height = DEControlHeight
        m_DataEntryControl.Width = DEControlWidth
    End If
    ERelations.Height = m_DataEntryControl.Height
    ERelations.Width = m_DataEntryControl.Width

    tabView.Width = ERelations.Width + 300
    tabView.Height = ERelations.Top + ERelations.Height - tabView.Top + 150
    ERelations.Left = tabView.Left + 150
    m_DataEntryControl.Left = ERelations.Left
    m_DataEntryControl.Top = ERelations.Top

    cmdCancel.Top = cmdOK.Top + cmdOK.Height + 75
    cmdOK.Left = tabView.Left + tabView.Width + 75
    cmdCancel.Left = cmdOK.Left

    MinimumHeight = tabView.Top + tabView.Height + 150
    MinimumWidth = cmdOK.Left + cmdOK.Width + BorderWidth + 150
    Me.Width = cmdOK.Left + cmdOK.Width + 150
    Me.Height = tabView.Top + tabView.Height + BorderHeight
    If Me.Height < MinimumHeight Then Me.Height = MinimumHeight
    CenterForm Me
    Err.Clear
End Sub

Private Sub Form_Initialize()
    Set Operation = New clsOperation
    CallUpdateRecordOperation = False
    Set m_Script = New ScriptControl
    m_Script.Language = "VBScript"
    m_CancelLoad = False
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    On Error Resume Next
    If (KeyCode = vbKeyM) And (Shift = vbAltMask) Then
        Select Case tabView.SelectedItem.Key
            Case "General"
            Case "RelatedInfo"
                ERelations.ShowMenu
            Case Else
                lstChild.ShowMenu
        End Select
    End If
    Err.Clear
End Sub

Private Sub Form_Paint()
    Me.Line (0, ScaleTop)-(ScaleWidth, ScaleHeight), BackColor, BF
End Sub
```

EXHIBIT F

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    Select Case UnloadMode
        Case vbFormCode
            Case Else
                Message.MessageType = MTCancel
                m_Cancel = True
                DoUnload Cancel
    End Select
End Sub

Private Sub Form_Resize()
    On Error Resume Next

    If Not m_FormLoaded Then Exit Sub

    tabView.Width = ScaleWidth - cmdOK.Width - 150
    tabView.Height = ScaleHeight - tabView.Top - 75
    ERelations.Left = tabView.Left + 150
    ERelations.Height = tabView.Height - (ERelations.Top - tabView.Top) - 150
    ERelations.Width = tabView.Width - 300
    If TypeName(m_DataEntryControl) = "DataEntryControl" Then
        m_DataEntryControl.Height = ERelations.Height
        m_DataEntryControl.Width = ERelations.Width
    End If
    cmdOK.Left = tabView.Left + tabView.Width + 75
    cmdCancel.Left = cmdOK.Left
    lstChild.Move ERelations.Left, ERelations.Top, ERelations.Width, ERelations.Height

    Err.Clear
End Sub

Private Sub Form_Terminate()
    Set m_DataEntryControl = Nothing
    Set Operation = Nothing
    Set m_Script = Nothing
    Set m_AppEnvironment = Nothing
End Sub

Private Sub Form_Unload(Cancel As Integer)
    On Error Resume Next
    m_DataEntryControl.UnloadControls
    Err.Clear
End Sub

Private Sub lstChild_ItemChanged(ItemEffectuated As Long)
    lstChild.Refresh
    lstChild.RowID = ItemEffectuated
End Sub

Private Sub lstChild_NewItem(ItemSelected As Long)
    lstChild.Refresh
    lstChild.RowID = ItemSelected
End Sub

Private Sub mnuEditCopy_Click()
    SendKeys "^C"
End Sub

Private Sub mnuEditCut_Click()
    SendKeys "^X"
End Sub

Private Sub mnuEditPaste_Click()
    SendKeys "^V"
End Sub

Private Sub mnuFileClose_Click()
    cmdOK_Click
End Sub

Public Sub RefreshForm()
    Dim Cancel As Integer

    m_FormLoaded = False
    lstChild.MultiSelect = True
    SetEntityObject
    SetFormSize
    m_FormLoaded = True
    tabView_Click
```


EXHIBIT F

```
LoadScripts
RaiseEventEx "FormLoad", m_TableName, Cancel
If Cancel Then m_CancelLoad = True
End Sub

Private Sub mnuOptionsDetail_Click(Index As Integer)
    Set m_Operations = GetMenuItemOperations(mnuOptionsDetail(Index).Tag)
End Sub

Private Sub Operation_DisplayDataEntryScreen(Message As MessageClass)
    Dim f As frmInput
    Set f = New frmInput
    Message.CurrentObjectID = MessageHandler.GetNextObjectID()
    Set f.Message = Message
    f.RefreshForm
    f.Show vbModal
    Set Message = f.Message
    Unload f
    Set f = Nothing
End Sub

Private Sub Operation_DisplaySearchScreen(LoadEntityID As Long, Message As AppUtils.MessageClass,
MultipleSelection As Boolean)
    Dim f As frmFind

    Set f = New frmFind
    Message.CurrentObjectID = MessageHandler.GetNextObjectID()
    f.EntityID = LoadEntityID
    f.Message = Message
    f.MultiSelect = MultipleSelection
    f.RefreshForm
    f.Show vbModal
    Unload f
    Set f = Nothing
End Sub

Private Sub tabView_Click()
    Dim Cancel As Integer

    Select Case tabView.SelectedItem.Key
        Case "General"
            ERelations.Visible = False
            lstChild.Visible = False
            m_DataEntryControl.Visible = True
            m_DataEntryControl.ZOrder
            m_FormLoaded = False
            SetFormSize
            m_FormLoaded = True
        Case "RelatedInfo"
            RaiseEventEx "BeforeValidate", m_TableName, Cancel
            If Cancel Then Exit Sub
            If Not m_DataEntryControl.Validate() Then
                tabView.Tabs(1).Selected = True
                Exit Sub
            End If
            RaiseEventEx "AfterValidate", m_TableName, Cancel
            If Cancel Then Exit Sub
            SaveEntity Cancel
            If Cancel Then Exit Sub
            ERelations.ParentEntityID = Message.EntityID
            ERelations.ParentRowID = Message.RowIDArray(0)
            lstChild.Visible = False
            m_DataEntryControl.Visible = False
            ERelations.Refresh
            ERelations.Visible = True
            ERelations.ZOrder
        Case Else
            RaiseEventEx "BeforeValidate", m_TableName, Cancel
            If Cancel Then Exit Sub
            If Not m_DataEntryControl.Validate() Then
                tabView.Tabs(1).Selected = True
                Exit Sub
            End If
            RaiseEventEx "AfterValidate", m_TableName, Cancel
            If Cancel Then Exit Sub
            SaveEntity Cancel
            If Cancel Then Exit Sub
            lstChild.ParentEntityID = Message.EntityID
            lstChild.ParentRowID = Message.RowIDArray(0)
```

EXHIBIT F

```
    1stChild.EntityID = CLng(Right$(tabView.SelectedItem.Key, Len(tabView.SelectedItem.Key) - 1))
    1stChild.Filter = GetAppInfo(1stChild.ParentEntityID, "ENTITY", "UNIQUE_FIELD_NAME") & " = " &
1stChild.ParentRowID
    1stChild.LoadStructureOnly = False
    1stChild.Context = gData.ApplicationInfo.GetRelationshipInfo(1stChild.ParentEntityID, 1stChild.EntityID,
"RELATION_TYPE")
    1stChild.Refresh
    m_DataEntryControl.Visible = False
    ERelations.Visible = False
    1stChild.Visible = True
End Select
End Sub

Private Sub SaveEntity(Optional Cancel As Integer = 0)
    Dim Result As Boolean
    Dim ID As Long

    RaiseEventEx "BeforeSave", m_TableName, Cancel
    If Cancel Then
        Exit Sub
    End If
    m_DataEntryControl.Save
    ID = CLng(m_DataEntryControl.GetCurrentID())
    Message.CurrentRowID = ID
    Message.RowIDArray = Array(ID)
    If m_DataEntryControl.NewRecord Then
        Result = CallByName(Operation, "opInsertRecord", VbMethod, Message)
        If Not Result Then
            Exit Sub
        End If
    ElseIf CallUpdateRecordOperation Then
        Result = CallByName(Operation, "opUpdateRecord", VbMethod, Message)
        If Not Result Then
            Exit Sub
        End If
    End If
    m_DataEntryControl.NewRecord = False
    RaiseEventEx "AfterSave", m_TableName
End Sub

Private Sub tbarOptions_ButtonClick(ByVal Button As MSComctlLib.Button)
    Select Case Button.Key
        Case "Cut"
            mnuEditCut_Click
        Case "Copy"
            mnuEditCopy_Click
        Case "Paste"
            mnuEditPaste_Click
    End Select
End Sub

Public Property Get FormControls() As Object
    On Error Resume Next
    If TypeName(m_DataEntryControl) = "DataEntryControl" Then
        Set FormControls = Me.Controls
    Else
        Set FormControls = m_DataEntryControl.Controls
        If Err.Number <> 0 Then
            Err.Clear
            Set FormControls = Me.Controls
        End If
    End If
End Property

Public Sub ShowDataEntryMenu(FieldName As String, ControlObjectID As Long)
    Dim i As Integer
    Dim Temp As String
    Dim ReferenceEntityID As Long
    Dim MenuItemGroupingID As Long

    ReferenceEntityID = GetEntityColumnInfo(Message.EntityID, FieldName, "REFERENCE_ENTITY_ID")

    m_DefaultMenuItem = LoadMenuItems("CONTROL_LIST", "LIST", mnuOptionsDetail, VItemArray)

    For i = 0 To mnuOptionsDetail.Count - 1
        With mnuOptionsDetail(i)
            If .Caption = "-" Then
                .Enabled = True
            Else
```

EXHIBIT F

```
If .Tag = vbNullString Then
    MenuItemGroupingID = -1
    .Enabled = False
Else
    MenuItemGroupingID = CLng(.Tag)
    .Enabled = IsValidMenuItem(ReferenceEntityID, gData.GroupID, MenuItemGroupingID, "CONTROL_LIST")
End If
End If
End With
Next i

For i = 0 To mnuOptionsDetail.Count - 1
    Temp = VItemArray(i)
    ReplaceVariables ReferenceEntityID, Temp
    If Temp <> "-" Then
        mnuOptionsDetail(i).Caption = Temp
    End If
Next i
PopupMenu mnuOptions
ExecuteOperations ReferenceEntityID, ControlObjectID
Set m_Operations = Nothing
End Sub

Private Sub ExecuteOperations(ReferenceEntityID As Long, ControlObjectID As Long)
    Dim O As Variant
    Dim Result As Boolean
    Dim M As MessageClass

    If Not m_Operations Is Nothing Then
        Set M = New MessageClass
        With M
            .CurrentRowID = -1
            .CurrentObjectID = -1
            .OwnerObjectID = ControlObjectID
            .Context = "CONTROL_LIST"
            .RowIDArray = Array(.CurrentRowID)
            .ParentEntityID = -1
            .ParentRowID = -1
            .EntityID = ReferenceEntityID
        End With

        For Each O In m_Operations
            Result = CallByName(Operation, "op" & O.OperationText, VbMethod, M)
            If Not Result Then
                Exit For
            End If
        Next O
        If Result Then
            M.MessageType = MTRefreshList
            MessageHandler.Broadcast Forms, M
        End If
    End If
End Sub

Private Sub ReplaceVariables(ReferenceEntityID As Long, Str As String)
    Dim Pos As Integer
    Dim Variable As String * 2
    Dim Temp As String

    Pos = 1
    Do
        Pos = InStr(Pos, Str, "@")
        If Pos > 0 Then
            Variable = Mid$(Str, Pos, 2)
            Select Case Variable

                Case "@E" 'Current entity
                    Temp = GetAppInfo(ReferenceEntityID, "ENTITY", "ENTITY_CAPTION")
                    Str = Replace(Str, Variable, Temp)

                Case "@P" 'Parent entity
                    Temp = GetAppInfo(-1, "ENTITY", "ENTITY_CAPTION")
                    Str = Replace(Str, Variable, Temp)

                Case Else
                    Str = Replace(Str, Variable, "????????")
            End Select
            Pos = Pos + 1
        End If
    Loop Until Pos = 0
```

EXHIBIT F

End Sub

```
Private Sub RaiseEventEx(EventName As String, TableName As String, Optional Cancel As Integer = 0)
    Dim CancelEvent As Variant
```

```
    On Error Resume Next
    Select Case EventName
        Case "AfterSave"
            m_Script.Run TableName & "_AfterSave"
        Case "AfterValidate"
            m_Script.Run TableName & "_AfterValidate", CancelEvent
        Case "BeforeSave"
            m_Script.Run TableName & "_BeforeSave", CancelEvent
        Case "BeforeValidate"
            m_Script.Run TableName & "_BeforeValidate", CancelEvent
        Case "FormLoad"
            m_Script.Run TableName & "_FormLoad", CancelEvent
        Case "FormUnload"
            m_Script.Run TableName & "_FormUnload", CancelEvent
    End Select
    Err.Clear
    Cancel = CancelEvent
End Sub
```

```
Private Sub LoadScripts()
```

```
    Dim c As Collection
    Dim i As Long
    Dim s As String
    Dim con As ADODB.Connection
```

```
    If TypeName(m_DataEntryControl) <> "DataEntryControl" Then Exit Sub
```

```
    Set c = GetEntityScripts(m_EntityID)
```

```
    If c Is Nothing Then Exit Sub
```

```
    For i = 1 To c.Count
```

```
        On Error Resume Next
```

```
        m_Script.AddCode c(i)
```

```
        Err.Clear
```

```
    Next i
```

```
    Set c = Nothing
```

```
    Set m_AppEnvironment = New CAppEnvironment
```

```
    m_AppEnvironment.EntityID = m_EntityID
```

```
    m_AppEnvironment.GridObject = m_DataEntryControl.GridObject
```

```
    Set c = GetEntityColumnScripts(m_EntityID)
```

```
    If c Is Nothing Then Exit Sub
```

```
    For i = 1 To c.Count
```

```
        On Error Resume Next
```

```
        m_Script.AddCode c(i)
```

```
        Err.Clear
```

```
    Next i
```

```
    Set c = Nothing
```

```
    Set con = New ADODB.Connection
```

```
    con.Open GetConnectionString(gData.ApplicationInfo.GetAppInfoValue(m_EntityID, "ENTITY", "DATASOURCE_ID"))
```

```
    Set m_AppEnvironment.DBConnection = con
```

```
    m_AppEnvironment.Refresh
```

```
    m_Script.AddObject "AppEnvironment", m_AppEnvironment, True
```

```
End Sub
```

```
Private Sub DoUnload(Optional Cancel As Integer = 0)
```

```
    RaiseEventEx "FormUnload", m_TableName, Cancel
```

```
    If Cancel Then
```

```
        Exit Sub
```

```
    End If
```

```
    Me.Hide
```

```
End Sub
```

EXHIBIT F

Data Entry Controls

```
Private m_EntityID As Long
Public NewRecord As Boolean
Public ReadOnly As Boolean
Public rsData As ADODB.Recordset
Public MessageHandler As MessageHandlerClass
Public Context As String
Private m_AppEnvironment As CAppEnvironment
Private m_Script As ScriptControl
Private m_KeyField As String
Private m_ParentKeyField As String
Private m_ParentEntityID As Long
Private m_ParentRowID As Long
Private m_HasParent As Boolean
Private m_Controls As Collection
Private m_ReferencedControls As Collection
Private m_CalculatedFields As Collection
Private m_PreventValidation As Boolean
```

```
Private Sub cboEdit_ButtonClick()
    ShowInfo grdData.Row, grdData.Columns(cboEdit.Tag).Index, True
End Sub
```

```
Private Sub grdData_AfterColUpdate(ByVal ColIndex As Integer)
```

```
    Dim cf As CCalculatedField
    Dim fieldName As String
    Dim vList As JSValueList
    Dim rs As ADODB.Recordset
    Dim col As JSColumn
    Dim ReferenceEntityID As Long
    Dim RecordSource As String
    Dim DisplayColumn As Variant
```

```
    On Error Resume Next
```

```
    fieldName = m_ReferencedControls(grdData.Columns(ColIndex).DataField)
```

```
    If Err.Number = 0 Then
```

```
        For Each cf In m_CalculatedFields
```

```
            On Error Resume Next
```

```
            fieldName = cf.ReferencedControls(fieldName)
```

```
            If Err.Number = 0 Then
```

```
                Select Case cf.ReturnType
```

```
                    Case "VALUE"
```

```
                        grdData.Value(cf.ColIndex) = EvalExpression(m_EntityID, grdData, cf)
```

```
                    Case "VALUE_LIST"
```

```
                        Set col = grdData.Columns(cf.ColIndex)
```

```
                        ReferenceEntityID = GetEntityColumnInfo(m_EntityID, col.DataField, "REFERENCE_ENTITY_ID")
```

```
                        RecordSource = FormatSQLExpression(m_EntityID, grdData, cf)
```

```
                        DisplayColumn = GetEntityColumnInfo(m_EntityID, col.DataField, "DISPLAY_COLUMN")
```

```
                        If IsNull(DisplayColumn) Then DisplayColumn = 1
```

```
                        Set rs = GetReportRS(ReferenceEntityID, RecordSource)
```

```
                        If Not rs Is Nothing Then
```

```
                            Set vList = col.ValueList
```

```
                            vList.Clear
```

```
                            Do While Not rs.EOF
```

```
                                vList.Add rs(0).Value, rs(CLng(DisplayColumn)).Value & ""
```

```
                                rs.MoveNext
```

```
                            Loop
```

```
                            rs.Close: Set rs = Nothing
```

```
                        End If
```

```
                        grdData.Value(cf.ColIndex) = Null
```

```
                    End Select
```

```
            End If
```

```
            Err.Clear
```

```
        Next cf
```

```
    End If
```

```
    Err.Clear
```

```
End Sub
```

```
Private Sub grdData_BeforeColUpdate(ByVal Row As Long, ByVal ColIndex As Integer, ByVal OldValue As String,
ByVal Cancel As GridEX20.JSRetBoolean)
```

```
    Dim Msg As String
```

```
    If Not m_PreventValidation Then
```

```
        m_PreventValidation = True
```

```
        Cancel = Not ValidValue(grdData, m_Controls, ColIndex, Msg)
```

```
        If Cancel Then
```

```
            LockWindow grdData.hwnd, False
```

EXHIBIT F

```
MsgBox Msg, vbCritical + vbOKOnly, gData.Caption
End If
m_PreventValidation = False
If Cancel Then Exit Sub
End If
End Sub

Private Sub grdData_CardResize(ByVal NewCardWidth As Long, ByVal Cancel As GridEX20.JSRetBoolean)
    picHeading.Width = NewCardWidth
End Sub

Private Sub grdData_ColButtonClick(ByVal ColIndex As Integer)
    Dim ControlType As String
    Dim v As Variant
    Dim FDC_Type As Variant
    Dim f As Form
    Dim Result As Boolean
    Dim i As Long

    ControlType = GetEntityColumnInfo(EntityID, grdData.Columns(ColIndex).DataField, "CONTROL_TYPE")
    Select Case ControlType
        Case "SmartComboControl"
            ShowFind ColIndex
        Case "FileDialogControl"
            FDC_Type = GetEntityColumnInfo(EntityID, grdData.Columns(ColIndex).DataField, "FDC_TYPE")
            If IsNull(FDC_Type) Then FDC_Type = "OPEN"
            FDC_Type = UCase$(FDC_Type)
            v = GetEntityColumnInfo(EntityID, grdData.Columns(ColIndex).DataField, "FDC_TITLE")
            If IsNull(v) Then
                v = IIf(FDC_Type = "OPEN", "Open File", "Save File")
            End If
            dlgPath.DialogTitle = v
            dlgPath.CancelError = True
            dlgPath.Flags = cdIOFNPathMustExist Or cdIOFNFileMustExist
            v = GetEntityColumnInfo(EntityID, grdData.Columns(ColIndex).DataField, "FDC_FILTER")
            If IsNull(v) Then
                v = "All Files (*.*)|*.*|Text Files (*.txt)|*.txt"
            End If
            dlgPath.Filter = v
            dlgPath.FilterIndex = 0
            On Error Resume Next
            If FDC_Type = "OPEN" Then
                dlgPath.ShowOpen
            Else
                dlgPath.ShowSave
            End If
            If Err.Number <> 0 Then
                Err.Clear
                Exit Sub
            End If
            grdData.Value(ColIndex) = dlgPath.FileName
        Case "NotesControl"
            Set f = New frmNotes
            f.Notes = grdData.Value(ColIndex)
            f.RefreshForm
            f.Show vbModal
            If Not f.Cancel Then
                grdData.Value(ColIndex) = f.Notes
            End If
            Unload f
            Set f = Nothing
    End Select
End Sub

Private Sub grdData_EndCustomEdit(ByVal ColIndex As Integer)
    Dim ControlType As String

    ControlType = GetEntityColumnInfo(EntityID, grdData.Columns(ColIndex).DataField, "CONTROL_TYPE")
    Select Case ControlType
        Case "ComboBoxControl"
            If (grdData.Value(ColIndex) & "") <> (cboEdit.Value & "") Then
                grdData.Value(ColIndex) = cboEdit.Value
            End If
    End Select
End Sub

Private Sub grdData_InitCustomEdit(ByVal ColIndex As Integer, ByVal EditBackColor As stdole.OLE_COLOR, ByVal EditForeColor As stdole.OLE_COLOR, ByVal EditFont As stdole.Font)
    Dim ControlType As String
```

EXHIBIT F

```
Dim AllowEdit As Boolean

ControlType = GetEntityColumnInfo(EntityID, grdData.Columns(ColIndex).DataField, "CONTROL_TYPE")
AllowEdit = GetEntityColumnInfo(EntityID, grdData.Columns(ColIndex).DataField, "ALLOW_EDIT")
Select Case ControlType
    Case "ComboBoxControl"
        Set cboEdit.ParentGrid = grdData
        cboEdit.ColIndex = ColIndex
        cboEdit.AllowEdit = AllowEdit
        cboEdit.Load grdData.Columns(ColIndex).ValueList
        cboEdit.Value = grdData.Value(ColIndex)
        cboEdit.Tag = grdData.Columns(ColIndex).DataField
    End Select
End Sub

Private Sub grdData_KeyPress(KeyAscii As Integer)
    If cboEdit.Visible Then cboEdit.PassKeyPress KeyAscii
End Sub

Private Sub grdData_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
    Dim i As Integer
    Dim ControlType As String
    Dim r As Long
    Dim c As Integer
    Dim rd As JSRowData

    r = 1

    c = -1
    Select Case x
        Case grdData.CellLeft(1, 1) To (grdData.CellLeft(1, 1) + grdData.CellWidth(1, 1))
        Case Else
            Exit Sub
        End Select
    For i = 1 To grdData.Columns.Count
        Select Case y
            Case grdData.CellTop(1, i) To (grdData.CellTop(1, i) + grdData.CellHeight(1, i))
                c = i
                Exit For
            Case Else
                /
        End Select
    Next i
    If c = -1 Then
        Exit Sub
    End If
    ControlType = GetEntityColumnInfo(m_EntityID, grdData.Columns(c).DataField, "CONTROL_TYPE") & ""
    If ControlType = "SmartComboControl" Then
        Set rd = grdData.GetRowData(r)
        If UserControl.TextWidth(rd.Value(c) & "") > (x - grdData.CellLeft(r, c)) Then
            ShowInfo r, c, False
        End If
    End If
End Sub

Private Sub grdData_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
    Dim i As Integer
    Dim ControlType As String
    Dim r As Long
    Dim c As Integer
    Dim rd As JSRowData

    r = 1
    c = -1
    Select Case x
        Case grdData.CellLeft(1, 1) To (grdData.CellLeft(1, 1) + grdData.CellWidth(1, 1))
        Case Else
            Screen.MouseIcon = LoadPicture("")
            Screen.MousePointer = vbDefault
            Exit Sub
        End Select
    For i = 1 To grdData.Columns.Count
        Select Case y
            Case grdData.CellTop(1, i) To (grdData.CellTop(1, i) + grdData.CellHeight(1, i))
                c = i
                Exit For
            Case Else
                /
        End Select
    Next i
    If c = -1 Then
```

EXHIBIT F

```
Screen.MouseIcon = LoadPicture("")
Screen.MousePointer = vbDefault
Exit Sub
End If
ControlType = GetEntityColumnInfo(m_EntityID, grdData.Columns(c).DataField, "CONTROL_TYPE") & ""
If ControlType = "SmartComboControl" Then
    Set rd = grdData.GetRowData(r)
    If UserControl.TextWidth(rd.Value(c) & "") > (x - grdData.CellLeft(r, c)) Then
        Screen.MouseIcon = LoadPicture(App.Path & "\Hand.cur")
        Screen.MousePointer = vbCustom
    Else
        Screen.MouseIcon = LoadPicture("")
        Screen.MousePointer = vbDefault
    End If
Else
    Screen.MouseIcon = LoadPicture("")
    Screen.MousePointer = vbDefault
End If
End Sub

Private Sub grdData_RowColChange(ByVal LastRow As Long, ByVal LastCol As Integer)
    Dim Cancel As Integer
    If grdData.col = 0 Then grdData.col = 1

    On Error Resume Next
    If grdData.col <> LastCol Then 'Field changed
        RaiseEventEx "LostFocus", grdData.Columns(LastCol).DataField, Cancel
        If Cancel Then
            grdData.col = LastCol
            Exit Sub
        End If
        RaiseEventEx "GotFocus", grdData.Columns(grdData.col).DataField
    End If
    Err.Clear
End Sub

Private Sub RaiseEventEx(EventName As String, FieldName As String, Optional Cancel As Integer = 0)
    Dim CancelEvent As Variant
    On Error Resume Next
    Select Case EventName
        Case "GotFocus"
            m_Script.Run FieldName & "_GotFocus"
        Case "LostFocus"
            m_Script.Run FieldName & "_LostFocus", CancelEvent
            Cancel = CancelEvent
    End Select
    Err.Clear
End Sub

Private Sub grdData_ShowCustomEdit(ByVal ColIndex As Integer, ByVal EditLeft As Single, ByVal EditTop As Single,
ByVal EditWidth As Single, ByVal EditHeight As Single, ByVal EditVisible As Boolean)
    Dim ControlType As String

    ControlType = GetEntityColumnInfo(EntityID, grdData.Columns(ColIndex).DataField, "CONTROL_TYPE")

    Select Case ControlType
        Case "ComboBoxControl"
            If EditVisible Then
                grdData.HideSelection = jgexHighLightNormal
                picBorder.Move grdData.Left + ScaleX(EditLeft, vbTwips, UserControl.ScaleMode), grdData.Top +
ScaleY(EditTop, vbTwips, UserControl.ScaleMode), ScaleX(EditWidth, vbTwips, UserControl.ScaleMode),
ScaleY(EditHeight, vbTwips, UserControl.ScaleMode)
                picBorder.Visible = True
                cboEdit.SetFocus
            Else
                grdData.SetFocus
                grdData.HideSelection = jgexHideSelection
                picBorder.Visible = False
            End If
        End Select
    End Sub

Private Sub grdData_UnboundReadData(ByVal RowIndex As Long, ByVal Bookmark As Variant, ByVal Values As
GridEX20.JSRowData)
    Dim i As Integer
    Dim cf As CCalculatedField
    Dim vList As JSValueList
    Dim rs As ADODB.Recordset
```


EXHIBIT F

```
Dim col As JSColumn
Dim ReferenceEntityID As Long
Dim RecordSource As String
Dim DisplayColumn As Variant

If rsData.BOF And rsData.EOF Then
    Exit Sub
End If

On Error Resume Next
i = grdData.Columns(m_KeyField).Index
If Err.Number <> 0 Then
    Err.Clear
    Exit Sub
End If

For i = 1 To Values.ColCount
    Set col = grdData.Columns(i)
    If NewRecord And i = grdData.Columns(m_KeyField).Index Then
        On Error Resume Next
        If IsNull(rsData.Fields(col.DataField).Value) Or IsEmpty(rsData.Fields(col.DataField).Value) Then
            Values(i) = "New Record"
        Else
            Values(i) = rsData.Fields(col.DataField).Value
        End If
        Err.Clear
    Else
        On Error Resume Next
        cf = m_CalculatedFields(col.DataField)
        If Err.Number = 0 Then
            If cf.IsBound Then
                On Error Resume Next
                Values(i) = rsData.Fields(col.DataField).Value
                Err.Clear
            End If
        Else
            On Error Resume Next
            Values(i) = rsData.Fields(col.DataField).Value
            Err.Clear
        End If
    End If
End If
Next i
'May not be necessary. Refer to BeforeColUpdate event.
If m_CalculatedFields Is Nothing Then Exit Sub
For Each cf In m_CalculatedFields
    Select Case cf.ReturnType
        Case "VALUE"
            If (Not cf.IsBound) Then
                Values(cf.ColIndex) = EvalExpression(m_EntityID, grdData, cf)
            End If
        Case "VALUE_LIST"
            Set col = grdData.Columns(cf.ColIndex)
            ReferenceEntityID = GetEntityColumnInfo(m_EntityID, col.DataField, "REFERENCE_ENTITY_ID")
            RecordSource = FormatSQLExpression(m_EntityID, grdData, cf)
            DisplayColumn = GetEntityColumnInfo(m_EntityID, col.DataField, "DISPLAY_COLUMN")
            If IsNull(DisplayColumn) Then DisplayColumn = 1
            Set rs = GetReportRS(ReferenceEntityID, RecordSource)
            If Not rs Is Nothing Then
                col.EditType = jgexEditCustom
                Set vList = col.ValueList
                vList.Clear
                Do While Not rs.EOF
                    vList.Add rs(0).Value, rs(CLng(DisplayColumn)).Value & ""
                    rs.MoveNext
                Loop
                rs.Close: Set rs = Nothing
            End If
        End Select
    End Select
Next cf
End Sub

Private Sub grdData_UnboundUpdate(ByVal RowIndex As Long, ByVal Bookmark As Variant, ByVal Values As GridEX20.JSRowData)
    Dim i As Integer
    Dim v As Variant
    Dim cf As CCalculatedField
    Dim col As JSColumn

    For i = 1 To Values.ColCount
```

EXHIBIT F

```
Set col = grdData.Columns(i)
If i <> grdData.Columns(m_KeyField).Index Then
    If IsEmpty(Values(i)) Then
        v = Null
    Else
        v = Values(i) & ""
        If Trim$(v) = "" Then v = Null
    End If
    If Not IsNull(v) Then
        Select Case UCase$(col.Format)
            Case "PERCENT"
                If InStr(1, v, "%") > 0 Then v = Val(v) / 100
        End Select
    End If
    On Error Resume Next
    cf = m_CalculatedFields(col.DataField)
    If Err.Number = 0 Then
        If cf.IsBound Then
            On Error Resume Next
            rsData.Fields(col.DataField).Value = v
            Err.Clear
        End If
    Else
        On Error Resume Next
        rsData.Fields(col.DataField).Value = v
        Err.Clear
    End If
End If
Next i
End Sub

Private Sub picBorder_Resize()
    cboEdit.Move -30, (picBorder.ScaleHeight - cboEdit.Height) \ 2, picBorder.ScaleWidth + 60
End Sub

Private Sub picHeading_Resize()
    On Error Resume Next
    Line1.X1 = picHeading.ScaleLeft
    Line1.X2 = picHeading.ScaleWidth
    Line1.Y1 = picHeading.ScaleHeight - (Line1.BorderWidth * 15)
    Line1.Y2 = picHeading.ScaleHeight - (Line1.BorderWidth * 15)
    lblCaption.Move picHeading.ScaleLeft, (picHeading.ScaleHeight - lblCaption.Height) / 2, picHeading.ScaleWidth
    ' , picHeading.ScaleHeight - (Line1.BorderWidth * 15)
    Err.Clear
End Sub

Private Sub UserControl_Initialize()
    m_HasParent = False
    grdData.RowHeight = 315
    Set m_Script = New ScriptControl
    m_Script.Language = "VBScript"
End Sub

Private Sub UserControl_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = vbKeyTab Then
        If cboEdit.Visible Then
            If grdData.col = grdData.Columns.Count Then
                grdData.col = 1
            Else
                grdData.col = grdData.col + 1
            End If
            KeyCode = 0
        End If
    End If
End Sub

Private Sub UserControl_Resize()
    On Error Resume Next
    grdData.Move 0, 0, ScaleWidth, ScaleHeight
    picHeading.Top = grdData.Top + 30
    picHeading.Left = grdData.Left + 30
    picHeading.Height = grdData.RowHeight - 30
    picHeading.Width = grdData.CardWidth
    Err.Clear
End Sub
```

EXHIBIT F

```
Public Sub OnControlClick(DataField As String, ControlObjectID As Long)
    On Error Resume Next
    UserControl.Parent.ShowDataEntryMenu DataField, ControlObjectID
    Err.Clear
End Sub

Public Sub Refresh()
    Dim fDef As ADO.Field
    Dim col As JSColumn
    Dim i As Long
    Dim DuplicateRecord As Boolean
    Dim cf As CCalculatedField
    Dim c As Collection

    m_PreventValidation = True
    grdData.TabKeyBehavior = jgexColumnNavigation
    m_KeyField = GetAppInfo(m_EntityID, "ENTITY", "UNIQUE_FIELD_NAME")

    grdData.Columns.Clear
    grdData.ItemCount = 1

    Set c = GetEntityColumns(m_EntityID)
    For i = 1 To c.Count
        On Error Resume Next
        Set fDef = rsData.Fields(c(i))
        If Err.Number = 0 Then
            Set col = grdData.Columns.Add(fDef.Name, jgexText, jgexEditTextBox, fDef.Name)
            col.DataField = fDef.Name
            Select Case GetGenericDatatype(fDef.Type)
                Case GenericDate
                    col.SortType = jgexSortTypeDateTime
                Case GenericMemo, GenericText
                    col.SortType = jgexSortTypeString
                Case GenericNumber
                    col.SortType = jgexSortTypeNumeric
            End Select
        Else
            Set col = grdData.Columns.Add(c(i), jgexText, jgexEditTextBox, c(i))
            col.DataField = c(i)
        End If
        Err.Clear
        col.ColPosition = i
    Next i

    FormatColumns m_ParentEntityID, grdData, m_EntityID, m_Controls

    Set m_CalculatedFields = LoadCalculatedControls(m_EntityID, grdData)
    If Not m_CalculatedFields Is Nothing Then
        For Each cf In m_CalculatedFields
            For i = 1 To cf.ReferencedControls.Count
                If m_ReferencedControls Is Nothing Then
                    Set m_ReferencedControls = New Collection
                End If
                On Error Resume Next
                m_ReferencedControls.Add cf.ReferencedControls(i), cf.ReferencedControls(i)
                Err.Clear
            Next i
        Next cf
    End If

    SetColumnProperties
    DoEvents
    If .NewRecord Then
        If rsData.BOF And rsData.EOF Then
            rsData.AddNew
            DuplicateRecord = False
        Else
            DuplicateRecord = True
        End If
        grdData.Value(grdData.Columns(m_KeyField).Index) = "New Record"
        If m_HasParent Then
            On Error Resume Next
            grdData.Value(grdData.Columns(m_ParentKeyField).Index) = m_ParentRowID
            rsData(m_ParentKeyField).Value = m_ParentRowID
            Err.Clear
        End If
        If Not DuplicateRecord Then SetDefaultValues
    End If
    grdData.AllowEdit = Not ReadOnly
```

EXHIBIT F

```
If ReadOnly Then
    grdData.BackColor = vbButtonFace
    grdData.BackColorHeader = vbButtonFace
    grdData.BackColorBkg = vbButtonFace
End If
'Set focus to first data entry cell.
grdData.Row = 1: grdData.col = 2
m_PreventValidation = False
LoadScripts
End Sub

Public Sub Save()
    If NewRecord Then
        rsData(m_KeyField).Value = NextSequenceVal(m_EntityID)
    End If
    On Error Resume Next
    LockWindow grdData.hwnd, True
    grdData.Update
    LockWindow grdData.hwnd, False
    Err.Clear
End Sub

Private Sub UserControl_Paint()
    Static RightX As Single
    Static RightY As Single

    Line (RightX - 2, 2)-(RightX - 2, RightY - 2), UserControl.BackColor, BF
    Line (RightX - 1, 2)-(RightX - 1, RightY - 2), UserControl.BackColor, BF

    RightX = ScaleWidth
    RightY = ScaleHeight

    Line (0, 0)-(ScaleWidth, 0), vbButtonShadow, BF
    Line (1, 1)-(ScaleWidth - 1, 1), vb3DHighlight, BF

    Line (0, 1)-(0, ScaleHeight), vbButtonShadow, BF
    Line (1, 1)-(1, ScaleHeight), vb3DHighlight, BF

    Line (ScaleWidth - 2, 2)-(ScaleWidth - 2, ScaleHeight - 2), vbButtonShadow, BF
    Line (ScaleWidth - 1, 2)-(ScaleWidth - 1, ScaleHeight - 2), vb3DHighlight, BF

    Line (2, ScaleHeight - 2)-(ScaleWidth - 2, ScaleHeight - 2), vbButtonShadow, BF
    Line (2, ScaleHeight - 1)-(ScaleWidth - 1, ScaleHeight - 1), vb3DHighlight, BF
End Sub

Public Function ValidData() As Boolean
    If RequiredFields(grdData) Then
        ValidData = ApplyRules(m_EntityID, grdData, IIf(NewRecord, "I", "U"))
    Else
        ValidData = False
    End If
End Function

Public Sub SetCurrentID(NewID As Variant)
    'Include this method to make interface consistent with custom data entry usercontrols.
End Sub

Public Function GetCurrentID() As Variant
    On Error Resume Next
    GetCurrentID = rsData(m_KeyField).Value
    If Err.Number <> 0 Then
        GetCurrentID = -1
        Err.Clear
    End If
End Function

Public Property Get EntityID() As Long
    EntityID = m_EntityID
End Property

Public Property Let EntityID(ByVal rhs As Long)
    m_EntityID = rhs
End Property

Public Sub UnloadControls()
    'Include this method to make interface consistent with custom data entry usercontrols.
End Sub

Public Sub SetAppInformationObject(AppInfo As Object)
```

EXHIBIT F

```
'Include this method to make interface consistent with custom data entry usercontrols.
End Sub

Public Sub SetParentID(FieldName As String, ID As Long)
    m_ParentKeyField = FieldName
    m_ParentRowID = ID
    m_HasParent = True
End Sub

Private Function RequiredFields(GridControl As GridEX) As Boolean
    Dim i As Long
    Dim col As JSColumn

    RequiredFields = True
    For i = 1 To GridControl.Columns.Count
        Set col = GridControl.Columns(i)
        If CBool(col.Tag) And Trim$(GridControl.Value(i) & "") = "" Then
            RequiredFields = False
            MsgBox col.Caption & " is a required field!", vbOKOnly + vbInformation, "Required Field Missing"
            GridControl.col = i
            GridControl.SetFocus
            Exit Function
        End If
    Next i
End Function

Public Property Get Controls() As Object
    Set Controls = UserControl.Controls
End Property

Public Sub GetCardSize(CardHeight As Long, CardWidth As Long)
    Dim i As Integer

    CardHeight = 0
    For i = 1 To grdData.Columns.Count
        CardHeight = CardHeight + grdData.CellHeight(1, i)
    Next i
    CardWidth = grdData.CardWidth * 1.1
    CardHeight = grdData.CardWidth * 1.1
End Sub

Private Sub UserControl_Terminate()
    Set m_Controls = Nothing
    Set m_ReferencedControls = Nothing
    Set m_CalculatedFields = Nothing
    Set m_Script = Nothing
    Set m_AppEnvironment = Nothing
End Sub

Private Sub SetDefaultValues()
    Dim col As JSColumn

    For Each col In grdData.Columns
        If IsNull(rsData(col.DataField)) And col.DataField <> m_KeyField Then
            grdData.Value(col.Index) = col.DefaultValue
        End If
    Next col
End Sub

Private Sub SetColumnProperties()
    Dim i As Long
    Dim col As JSColumn

    For i = 1 To grdData.Columns.Count
        Set col = grdData.Columns(i)
        If col.DataField = m_KeyField Then
            col.EditType = jgexEditNone
            col.Visible = True
            col.CardCaption = True
        End If
    Next i
End Sub

Private Sub ShowInfo(RowIndex As Long, ColIndex As Integer, NewRecord As Boolean)
    Dim M As MessageClass
    Dim ReferenceEntityID As Long
    Dim f As frmInput
    Dim rs As ADODB.Recordset
    Dim ReferenceTable As Boolean
```

EXHIBIT F

```
Dim v As Variant
Dim ValidOperation As Boolean

If Not NewRecord Then
    grdData.col = ColIndex
    grdData.Row =RowIndex
End If
Set M = New MessageClass
ReferenceEntityID = GetEntityColumnInfo(m_EntityID, grdData.Columns(ColIndex).DataField,
"REFERENCE_ENTITY_ID")
ReferenceTable = GetAppInfo(ReferenceEntityID, "ENTITY", "REFERENCE")
v = IIf(ReferenceTable, "REFERENCE", "ENTITY")
If NewRecord Then
    ValidOperation = IsValidOperation(ReferenceEntityID, gData.GroupID, v & "", "DataEntryNew")
    If Not ValidOperation Then Exit Sub
    Set rs = GetEmptyRS(ReferenceEntityID, True)
Else
    ValidOperation = IsValidOperation(ReferenceEntityID, gData.GroupID, v & "", "DataEntryEdit")
    Set rs = GetSingleRowRS(ReferenceEntityID, CLng(grdData.Value(ColIndex)), True)
End If

With M
    .CurrentRowID = IIf(NewRecord, -1, grdData.Value(ColIndex))
    .CurrentObjectID = -1
    .OwnerObjectID = -1
    .RowIDArray = Array(.CurrentRowID)
    .ParentEntityID = -1
    .ParentRowID = -1
    .EntityID = ReferenceEntityID
    .Context = v
    If NewRecord Then
        .MessageType = MTNewRecord
    Else
        If ValidOperation Then
            .MessageType = MTEditRecord
        Else
            .MessageType = MTViewRecord
        End If
    End If
    Set .CurrentRS = rs
End With
Set f = New frmInput
Set f.Message = M
f.CallUpdateRecordOperation = True
f.RefreshForm
f.Show vbModal
Set M = f.Message
Unload f
Set f = Nothing
rs.Close: Set rs = Nothing
If M.MessageType <> MTCancel And NewRecord Then
    LoadValueList -1, m_EntityID, grdData.Columns(ColIndex)
    cboEdit.Load grdData.Columns(ColIndex).ValueList
    cboEdit.Value = M.CurrentRowID
End If
Set M = Nothing
End Sub

Private Sub ShowFind(ColIndex As Integer)
    Dim M As MessageClass
    Dim ReferenceEntityID As Long
    Dim ReferenceTable As Boolean
    Dim f As frmFind

    Set M = New MessageClass
    ReferenceEntityID = GetEntityColumnInfo(m_EntityID, grdData.Columns(ColIndex).DataField,
"REFERENCE_ENTITY_ID")
    ReferenceTable = GetAppInfo(ReferenceEntityID, "ENTITY", "REFERENCE")

    With M
        .CurrentRowID = -1
        .CurrentObjectID = -1
        .OwnerObjectID = -1
        .RowIDArray = Array(.CurrentRowID)
        .ParentEntityID = -1
        .ParentRowID = -1
        .EntityID = ReferenceEntityID
        If ReferenceTable Then
            .Context = "REFERENCE"
        End If
    End With
End Sub
```

EXHIBIT F

```
Else
    .Context = "ENTITY"
End If
.MessageType = MTFindRecord
End With

Set f = New frmFind
f.EntityID = M.EntityID
f.Message = M
Set M = Nothing
f.MultiSelect = False
f.RefreshForm
f.Show vbModal
Set M = f.Message
If M.MessageType <> MTCancel Then
    grdData.Value(ColIndex) = M.SelectedRows(0)
End If
Unload f
Set f = Nothing

End Sub

Private Function LoadColumnValue(ColIndex As Integer, Value As Variant) As Variant
    Dim ReferenceEntityID As Long
    Dim ControlType As String
    Dim rs As ADODB.Recordset
    Dim col As JSColumn
    Dim s As String
    Dim i As Long

    Set col = grdData.Columns(ColIndex)
    ControlType = GetEntityColumnInfo(EntityID, col.DataField, "CONTROL_TYPE")
    Select Case ControlType
        Case "SmartComboControl"
            ReferenceEntityID = GetEntityColumnInfo(EntityID, col.DataField, "REFERENCE_ENTITY_ID")
            Set rs = GetSingleRowRS(ReferenceEntityID, CLng(Value), False)
            col.MinRowsInCardView = rs.Fields.Count
            col.MaxRowsInCardView = col.MinRowsInCardView
            col.WordWrap = True
            s = ""
            For i = 0 To rs.Fields.Count - 1
                If s <> "" Then s = s & vbCrLf
                s = s & rs.Fields(i).Name & ": " & rs.Fields(i).Value & " "
            Next i
            rs.Close: Set rs = Nothing
            LoadColumnValue = s
        Case Else
            LoadColumnValue = Value
    End Select
End Function

Public Property Get ParentEntityID() As Long
    ParentEntityID = m_ParentEntityID
End Property

Public Property Let ParentEntityID(ByVal rhs As Long)
    m_ParentEntityID = rhs
End Property

Private Sub LoadScripts()
    Dim c As Collection
    Dim i As Long
    Dim s As String
    Dim con As ADODB.Connection

    Set m_AppEnvironment = New CAppEnvironment
    m_AppEnvironment.EntityID = m_EntityID
    m_AppEnvironment.GridObject = grdData
    Set c = GetEntityColumnScripts(m_EntityID)
    If c Is Nothing Then Exit Sub
    For i = 1 To c.Count
        On Error Resume Next
        m_Script.AddCode c(i)
        Err.Clear
    Next i
    Set c = Nothing
    Set con = New ADODB.Connection
    con.Open GetConnectionString(gData.ApplicationInfo.GetAppInfoValue(m_EntityID, "ENTITY", "DATASOURCE_ID"))
    Set m_AppEnvironment.DBConnection = con
```

EXHIBIT F

```
m_AppEnvironment.Refresh
m_Script.AddObject "AppEnvironment", m_AppEnvironment, True
End Sub

Public Property Get GridObject() As GridEX
    Set GridObject = grdData
End Property
```


EXHIBIT F

Entity Business Rules

```
Public Function ApplyRules(EntityID As Long, ControlsCollection As Object, UpdateType As String) As Boolean
    Dim c As Collection
    Dim i As Integer
    Dim ER As ENTITY_RULE_T
    Dim Result As RuleActionType
    Dim GoToRuleNumber As Long

    ApplyRules = True
    Set c = AppInformation.GetEntityRules(EntityID, UpdateType)
    If c Is Nothing Then
        Exit Function
    End If
    GoToRuleNumber = -1
    For i = 1 To c.Count
        If (GoToRuleNumber = -1) Or (GoToRuleNumber = i) Then
            ER = c(i)
            If ER.ActionType = "STOP" Then Exit For
            If TypeName(ControlsCollection) = "GridEX" Then
                Result = ValidRuleGrid(EntityID, ER, ControlsCollection)
            Else
                Result = ValidRule(EntityID, ER, ControlsCollection)
            End If
            If Result = RuleFailed Then
                ApplyRules = False
                Exit For
            ElseIf Result = GoToRule Then
                GoToRuleNumber = ER.ActionValue
            Else
                GoToRuleNumber = -1
            End If
        End If
    Next i
End Function

Private Function ValidRule(EntityID As Long, EntityRule As ENTITY_RULE_T, ControlsCollection As Object) As RuleActionType
    Dim s As String
    Dim ControlName As String
    Dim FieldName As String
    Dim i As Integer
    Dim J As Integer
    Dim FieldList() As String
    Dim FieldListSize As Integer
    Dim v As Variant
    Dim rs As ADODB.Recordset
    Dim KeyField As String
    Dim DataType As GenericDatatype

    ValidRule = True

    KeyField = GetAppInfo(EntityID, "ENTITY", "UNIQUE_FIELD_NAME")

    s = EntityRule.RuleValue

    i = InStr(s, "{")
    FieldListSize = -1
    Do While i > 0
        J = InStr(i + 1, s, "}")
        FieldName = Mid$(s, i + 1, J - i - 1)
        FieldListSize = FieldListSize + 1
        ReDim Preserve FieldList(FieldListSize)
        FieldList(FieldListSize) = FieldName
        i = InStr(i + 1, s, "{")
    Loop

    For i = 0 To FieldListSize
        On Error Resume Next
        ControlName = GetControlName(FieldList(i), ControlsCollection)
        v = ControlsCollection(ControlName).Value
        If Err.Number <> 0 Then
            v = Null
            Err.Clear
        End If
        If FieldList(i) = KeyField Then
            If v = "New Record" Or IsNull(v) Then v = -1
        End If
    Next i
End Function
```

EXHIBIT F

```
J = InStr(1, s, "{" & FieldList(i) & "}")

DataType = DB.GetDatatype(EntityID, FieldList(i))
v = DB.FixSQLValue(EntityID, v, Null, DataType, oeEQ)

s = Left$(s, J - 1) & (v & "") & Mid$(s, J + (Len(FieldList(i)) + 2))
Next i

Select Case EntityRule.RuleType
Case "SQL"
Set rs = DB.GetRuleRS(EntityID, s)
Case "STORED PROCEDURE"
Set rs = DB.GetRuleRS(EntityID, s, True)
End Select

If rs Is Nothing Then
MsgBox "Invalid business rule! Please refer to the system configuration database.", vbCritical + vbOKOnly,
"Invalid Business Rule"
ValidRule = RuleFailed
Exit Function
End If
If Not (rs.BOF And rs.EOF) Then
Select Case EntityRule.ActionType
Case "FATAL"
MsgBox EntityRule.ActionValue, vbCritical + vbOKOnly, "Fatal Error"
ValidRule = RuleFailed
Case "WARNING"
If MsgBox(EntityRule.ActionValue, vbInformation + vbYesNo, "Warning") = vbNo Then
ValidRule = RuleFailed
Else
ValidRule = RulePassed
End If
Case "GO_TO_RULE"
ValidRule = GoToRule
End Select
End If
rs.Close: Set rs = Nothing
End Function
```

EXHIBIT F

Entity Triggers

```
Public Function ApplyTriggers(EntityID As Long, RowID As Long, UpdateType As String) As Boolean
    ApplyTriggers = DB.ApplyTriggers(EntityID, RowID, UpdateType)
End Function
```

```
Public Function ApplyTriggers(EntityID As Long, RowID As Long, UpdateType As String) As Boolean
    Dim c As Collection
    Dim i As Integer
    Dim ER As ENTITY_RULE_T
    Dim Result As RuleActionType

    ApplyTriggers = True
    Set c = AppInformation.GetEntityRules(EntityID, UpdateType, True)
    If c Is Nothing Then
        Exit Function
    End If

    For i = 1 To c.Count
        ER = c(i)
        Result = ApplyTrigger(EntityID, ER, RowID)
        If Result = RuleFailed Then
            ApplyTriggers = False
            Exit For
        End If
    Next i

End Function
```

```
Private Function ApplyTrigger(EntityID As Long, EntityRule As ENTITY_RULE_T, RowID As Long) As RuleActionType
    Dim s As String
    Dim FieldName As String
    Dim i As Integer
    Dim J As Integer
    Dim FieldList() As String
    Dim FieldListSize As Integer
    Dim v As Variant
    Dim rsParams As ADODB.Recordset
    Dim KeyField As String
    Dim DataType As GenericDatatype
    Dim SuccessfulQuery As Boolean

    ApplyTrigger = RulePassed

    KeyField = AppInformation.GetAppInfoValue(EntityID, "ENTITY", "UNIQUE_FIELD_NAME")
    s = EntityRule.RuleValue

    i = InStr(s, "{")
    FieldListSize = -1
    Do While i > 0
        J = InStr(i + 1, s, "}")
        FieldName = Mid$(s, i + 1, J - i - 1)
        FieldListSize = FieldListSize + 1
        ReDim Preserve FieldList(FieldListSize)
        FieldList(FieldListSize) = FieldName
        i = InStr(i + 1, s, "{")
    Loop

    If FieldListSize <> -1 Then
        Set rsParams = GetSingleRowRS(EntityID, RowID, True)
        If rsParams Is Nothing Then
            ApplyTrigger = RuleFailed
            Exit Function
        End If
    End If

    For i = 0 To FieldListSize
        On Error Resume Next
        If IsEmpty(rsParams(FieldList(i)).Value) Then
            v = Null
        Else
            v = rsParams(FieldList(i)).Value
        End If
        If Err.Number <> 0 Then
            v = Null
            Err.Clear
        End If
    Next i
End Function
```

EXHIBIT F

```
J = InStr(1, s, "{" & FieldList(i) & "}")
DataType = DB.GetDatatype(EntityID, FieldList(i))
v = DB.FixSQLValue(EntityID, v, Null, DataType, oeEQ)
s = Left$(s, J - 1) & (v & "") & Mid$(s, J + (Len(FieldList(i)) + 2))
Next i

If Not rsParams Is Nothing Then
    rsParams.Close: Set rsParams = Nothing
End If
SuccessfulQuery = False
Select Case EntityRule.RuleType
    Case "SQL"
        SuccessfulQuery = ExecuteRule(EntityID, s, False)
    Case "STORED PROCEDURE"
        SuccessfulQuery = ExecuteRule(EntityID, s, True)
End Select

If SuccessfulQuery Then
    ApplyTrigger = RulePassed
Else
    ApplyTrigger = RuleFailed
End If
End Function

Private Function ExecuteRule(EntityID As Long, SQL As String, Optional IsStoredProcedure As Boolean = False) As Boolean
    Dim com As ADODB.Command
    Dim dDef As ADOX.Catalog
    Dim i As Long
    Dim a As Variant
    Dim s As String

    On Error Resume Next
    If IsStoredProcedure Then
        i = InStr(1, SQL, " ", vbTextCompare)
        If i > 0 Then
            Set dDef = New ADOX.Catalog

            Set dDef.ActiveConnection = GetConnection(GetDatasourceID(EntityID))

            s = Right$(SQL, Len(SQL) - i)
            a = Split(s, ",", -1, vbTextCompare)
            SQL = Trim$(Left$(SQL, i - 1))
            Set com = dDef.Procedures(SQL).Command

            For i = 0 To UBound(a)
                com.Parameters(i).Value = a(i)
            Next i
        Else
            Set com = New ADODB.Command
            Set com.ActiveConnection = GetConnection(GetDatasourceID(EntityID))
            Set com = dDef.Procedures(SQL).Command
        End If
    Else
        Set com = New ADODB.Command
        Set com.ActiveConnection = GetConnection(GetDatasourceID(EntityID))

        If com.ActiveConnection Is Nothing Then
            Set com = Nothing
            ExecuteRule = False
            Exit Function
        End If
        com.CommandType = adCmdText
        com.CommandText = SQL
        com.CommandTimeout = AppInformation.GetAppInfoValue("QUERY_TIMEOUT", "SYSTEM_INFO", "ATTRIBUTE_VALUE")
    End If

    com.Execute
    ExecuteRule = (Err.Number = 0)
    Err.Clear
    Set com = Nothing
    Set dDef = Nothing
End Function
```